Using CPU Interrupts in the IsoPod

This applies to IsoPod kernel v0.38 and later.

Interrupt Vectors in Flash ROM

The DSP56F805 processor used in the IsoPod supports 64 interrupt vectors, in the first 128 locations of Flash ROM. Each vector is a two-word machine instruction, normally a JMP instruction to the corresponding interrupt routine. When an interrupt occurs, the CPU jumps directly to the appropriate address ($00-$7E) in the vector table.

Since this vector table is part of the IsoPod kernel, it cannot be altered by the user. Also, some interrupts are required for the proper functioning of the IsoPod, and these vectors must never be changed. So the IsoPod includes a "user" vector table at the high end of Flash ROM (addresses $7D80-7DFE). This is exactly the same as the "kernel" vector table, except that certain "reserved for IsoPod" interrupts have been excluded. The user vector table can be programmed, erased, and reprogrammed freely by the user, as long as suitable precautions are taken.

Writing Interrupt Service Routines

Interrupt service routines must be written in DSP56F805 machine language, and must end with an RTI (Return from Interrupt) instruction. Some peripherals will have additional requirements; for example, many interrupt sources need to be explicitly cleared by the interrupt service routine. For more information about interrupt service routines, refer to the Motorola DSP56800 16-Bit Digital Signal Processor Family Manual (Chapter 7), and the Motorola DSP56F801/803/805/807 16-Bit Digital Signal Processor User's Manual.

You should be aware that the IsoPod uses certain channels in the Interrupt Priority controller:

> The IsoMax Timer (Timer D3) is assigned to Interrupt Priority Channel 3.
> SCI#0 (RS-232) serial I/O is assigned to Interrupt Priority Channel 4.
> The I/O Scheduling Timer[1] is assigned to Interrupt Priority Channel 5.

These channels may be shared by other peripherals. However, it is important to remember that these channels are *enabled* by the IsoMax kernel after a reset, and must never be disabled. You should not use the corresponding bits in the Interrupt Priority Register as interrupt enable/disable bits.

Interrupt channels 0, 1, 2, and 6 are reserved for your use. The IsoMax kernel does not use them, and you may assign, enable, or disable them freely. Channel 0 has the lowest priority, and 6 the highest.[2]

The User Interrupt Vector Table

The user vector table is identical to the kernel (CPU) vector table, except that it starts at address $7D80 instead of address $0. Each interrupt vector is two words in this table, sufficient for a machine language jump instruction. For all interrupts which are not reserved by IsoMax, the kernel vector table simply jumps to the corresponding location in the user vector table. (Remember that this adds the overhead of one absolute jump instruction -- 6 machine clock cycles -- to the interrupt service.)

Note: IsoPod kernels version 0.37 and earlier do not support a user vector table.

Note: This table is subject to change. Future versions of the IsoPod software may reserve more of these interrupts for internal use, as more I/O functions are added to the IsoPod kernel.

---

[1] This will be a feature of future IsoMax kernels. Interrupt channel 5 is reserved for this use.

[2] Use channel 6 only for critically-urgent interrupts, since it will take priority over channels 4 and 5, both of which require prompt service.

Interrupt Number	User Vector Address	Kernel Vector Address	Description
0		$00	reset - *reserved for IsoPod*
1	$7D82	$02	COP Watchdog reset
2	$7D84	$04	reserved by Motorola
3		$06	illegal instruction - *reserved for IsoPod*
4	$7D88	$08	Software interrupt
5	$7D8A	$0A	hardware stack overflow
6	$7D8C	$0C	OnCE Trap
7	$7D8E	$0E	reserved by Motorola
8	$7D90	$10	external interrupt A
9	$7D92	$12	external interrupt B
10	$7D94	$14	reserved by Motorola
11	$7D96	$16	boot flash interface
12	$7D98	$18	program flash interface
13	$7D9A	$1A	data flash interface
14	$7D9C	$1C	MSCAN transmitter ready
15	$7D9E	$1E	MSCAN receiver full
16	$7DA0	$20	MSCAN error
17	$7DA2	$22	MSCAN wakeup
18	$7DA4	$24	reserved by Motorola
19	$7DA6	$26	GPIO E
20	$7DA8	$28	GPIO D
21	$7DAA	$2A	reserved by Motorola
22	$7DAC	$2C	GPIO B
23	$7DAE	$2E	GPIO A
24	$7DB0	$30	SPI transmitter empty
25	$7DB2	$32	SPI receiver full/error
26	$7DB4	$34	Quad decoder #1 home
27	$7DB6	$36	Quad decoder #1 index pulse
28	$7DB8	$38	Quad decoder #0 home
29	$7DBA	$3A	Quad decoder #0 index pulse
30	$7DBC	$3C	Timer D Channel 0
31	$7DBE	$3E	Timer D Channel 1
32	$7DC0	$40	Timer D Channel 2
33		$42	Timer D Channel 3 - *reserved for IsoPod*
34	$7DC4	$44	Timer C Channel 0
35	$7DC6	$46	Timer C Channel 1
36	$7DC8	$48	Timer C Channel 2
37	$7DCA	$4A	Timer C Channel 3
38	$7DCC	$4C	Timer B Channel 0
39	$7DCE	$4E	Timer B Channel 1
40	$7DD0	$50	Timer B Channel 2
41	$7DD2	$52	Timer B Channel 3
42	$7DD4	$54	Timer A Channel 0
43	$7DD6	$56	Timer A Channel 1
44	$7DD8	$58	Timer A Channel 2
45	$7DDA	$5A	Timer A Channel 3
46	$7DDC	$5C	SCI #1 Transmit complete
47	$7DDE	$5E	SCI #1 transmitter ready
48	$7DE0	$60	SCI #1 receiver error
49	$7DE2	$62	SCI #1 receiver full
50	$7DE4	$64	SCI #0 Transmit complete

Interrupt Number	User Vector Address	Kernel Vector Address	Description
51		$66	SCI #0 transmitter ready - *reserved for IsoPod*
52	$7DE8	$68	SCI #0 receiver error
53		$6A	SCI #0 receiver full - *reserved for IsoPod*
54	$7DEC	$6C	reserved by Motorola
55	$7DEE	$6E	ADC A Conversion complete
56	$7DF0	$70	reserved by Motorola
57	$7DF2	$72	ADC A zero crossing/error
58	$7DF4	$74	Reload PWM B
59	$7DF6	$76	Reload PWM A
60	$7DF8	$78	PWM B Fault
61	$7DFA	$7A	PWM A Fault
62	$7DFC	$7C	PLL loss of lock
63	$7DFE	$7E	low voltage detector

Clearing the User Vector Table

Since the user vector table is at the high end of Flash ROM, it will be erased by the SCRUB command (which erases all of the user-programmable Flash ROM).

If you wish to erase *only* the user vector table, you should use the command

```
HEX 7D00 PFERASE
```

This will erase 256 words of Program Flash ROM, starting at address 7D00. In other words, this will erase locations 7D00-7DFF, which includes the user vector table. Because of the limitations of Flash ROM, you cannot erase a smaller segment -- you must erase 256 words. However, this is at the high end of Flash ROM and is unlikely to affect your application program, which is built upward from low memory.

When Flash ROM is erased, all locations read as $FFFF. This is an illegal CPU instruction. So it is very important that you install an interrupt vector *before* you enable the corresponding interrupt! If you enable a peripheral interrupt when no vector has installed, you will cause an Illegal Instruction trap and the IsoPod will reset.[3]

Installing an Interrupt Vector

Once the Flash ROM has been erased, you can write data to it with the `PF!` operator. Each location can be written only once, and must be erased before being written with a different value.[4]

For example, this will program the low-voltage-detect interrupt to jump to address zero. (This will restart the IsoPod, since address zero is the reset address.)

```
HEX  E984 7DFE PF!  0 7DFF PF!
```

E984 is the machine language opcode for an absolute jump; this is written into the first word of the vector. The destination address, 0, is written into the second word. Because these addresses are in Flash ROM, you must use the `PF!` operator. An ordinary `!` operator will not work.

[3] This is why the "illegal instruction" interrupt is reserved for IsoMax. If it were vectored to the user table, and you did not install a vector for it, the attempt to service an illegal instruction would cause yet another illegal instruction, and the CPU would lock up.

[4] Strictly speaking, you can write a Flash ROM location more than once, but you can only change "1" bits to "0." Once a bit has been written as "0", you need to erase the ROM page to return it to a "1" state.

Precautions when using Interrupts

1. An unprogrammed interrupt vector will contain an FFFF instruction, which is an illegal instruction on the DSP56F805. Don't enable an interrupt until *after* you have installed its interrupt vector.

2. Remember that most interrupts must be cleared at the source before your service routine Returns from Interrupt (with an RTI instruction). If you forget to clear the interrupt, you may end in an infinite loop.

3. Remember that `SCRUB` will erase all vectors in the user table. Be sure to disable *all* of the interrupts that you have enabled, before you use `SCRUB`.

4. You cannot erase a single vector in the user table. You must use `HEX 7D00 PFERASE` to erase the entire table. As with `SCRUB`, be sure to disable all of your interrupt sources first.

5. Do *not* use the global interrupt enable (bits I1 and I0 in the Status Register) to disable your peripheral interrupts. This will also shut off the interrupts that are used by IsoMax, and the IsoPod will likely halt.

6. It *is* permissible to disable interrupts globally for extremely brief periods -- on the order of a few machine instructions -- in order to perform operations that mustn't be interrupted. But this may affect critical timing within IsoMax, and is generally discouraged.

7. You can perform the action of an IsoPod reset by jumping to absolute address zero. But note that, unlike a true hardware reset, this will *not* disable any interrupt sources that you may have enabled.